CodeGraph

Плейбук CodeGraph для разработки с ИИ-агентами

От продуктовой задачи до проверенного выпуска

ПРАКТИЧЕСКИЙ ПЛЕЙБУК

OpenSpec · РБПО · Release · Incident

Financial Control · DORA · Evidence

Публикационная редакция · 2026

Содержание

От продуктовой задачи до проверенного выпуска

> ИИ-агенты умеют за часы выполнять работу, на которую раньше уходили дни: разбирать требования, искать нужный код, предлагать архитектуру, писать реализацию и тесты. Но ускорение отдельного разработчика ещё не означает ускорение продукта. Узкое место смещается в постановку задачи, проверку результата, согласование рисков и выпуск.

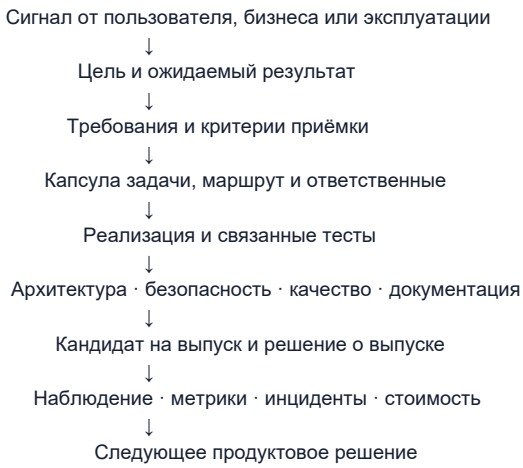
CodeGraph строит вокруг ИИ-агентов управляемый жизненный цикл:

- продуктовая задача превращается в проверяемые требования;
- требования связываются с конкретной работой;
- люди и цифровые роли получают ограниченные полномочия;
- изменение проверяется по коду, тестам, архитектуре и безопасности;
- решение о выпуске опирается на один пакет подтверждений;
- результаты эксплуатации возвращаются в продуктовый цикл.

Этот подход развивает идеи нескольких известных плейбуков. Anthropic предлагает завершать каждую стадию машиночитаемым артефактом. AWS строит совместную работу команды и ИИ вокруг плана, уточнений и человеческих решений, а в адаптивном продолжении методики меняет глубину процесса под сложность задачи. Сбер разделяет контур намерения и контур исполнения, дополняя их сквозной проверкой и управлением. Т-Банк начинает с общей платформы, отдельных сценариев и измерения командного эффекта.

Особенность CodeGraph — сквозная связь продуктового замысла с фактическим устройством системы. Платформа соединяет требования, задачи, код, тесты, риски, выпуск и эксплуатационные события. Граф свойств кода показывает, какие функции, сервисы и потоки данных затронуты изменением. Цифровые роли собирают подтверждения по своим областям ответственности. Итоговое решение остаётся за назначенным сотрудником.

Плейбук в одной схеме



Через весь цикл проходят три слоя:

1. **Контекст.** Репозитории, требования, архитектура, правила, история решений и состояние сред.
2. **Полномочия.** Кто может читать, предлагать, изменять, проверять, выпускать и принимать риск.
3. **Подтверждения.** Какой результат получен, к какой версии он относится и кто его принял.

Git, трекер, CI/CD, документы и системы эксплуатации сохраняют свои исходные записи. CodeGraph связывает их в общий контур и показывает, на чём основан текущий статус проекта.

1. Семь принципов CodeGraph

1. Результат важнее запроса к модели

Работа начинается с изменения, которое должно произойти для пользователя, оператора или бизнеса. Формулировка «добавить кнопку» описывает решение. Формулировка «сократить время оформления заявки с пяти до двух минут» задаёт проверяемый результат.

Чем быстрее ИИ выполняет реализацию, тем важнее качество исходной постановки. В методологии Сбера эта мысль выражена принципом «исследование задачи важнее запроса к модели». В CodeGraph она закрепляется через продуктовую цель, требования и критерии приёмки.

2. У каждого изменения есть сквозная история

Для любого результата должна восстанавливаться цепочка:

зачем сделали
→ что требовалось
→ кто выполнял
→ какой код изменился
→ какие проверки прошли
→ какие риски остались
→ кто разрешил выпуск
→ что произошло после выпуска

Такой подход близок к артефактной цепочке Anthropic, но CodeGraph дополняет её связями с графом кода, ролевыми заключениями и эксплуатационными событиями.

3. ИИ работает в пределах конкретной роли

Цифровая роль описывает конкретную ответственность: сформулировать требования, изменить код, проверить архитектуру, найти угрозы, провести тестирование, подготовить выпуск или посчитать затраты.

Для роли задаются:

- входные данные;
- разрешённые действия;
- ожидаемый результат;
- условия остановки;
- сотрудник, который принимает работу.

Имя цифрового сотрудника делает маршрут понятнее, но полномочия определяются настройками проекта и правами доступа.

4. Автор и проверяющий разделены

Агент, который написал изменение, передаёт его независимым контурам проверки. Архитектура, безопасность, тестирование, документация и эксплуатационная готовность рассматривают результат со своих точек зрения.

Это снижает зависимость от самооценки модели и делает каждое заключение самостоятельным.

5. Глубина процесса зависит от риска

Исправление текста, изменение внутренней функции и миграция данных требуют разной глубины процесса. CodeGraph назначает роли и проверки под конкретный тип работы.

На практике удобно использовать три маршрута:

Маршрут	Для каких изменений	Минимальный состав
Быстрый	Документация, локальное безопасное исправление	Требование, изменение, тест или проверка, независимое принятие
Основной	Дефект, функция, изменение интерфейса внутри продукта	Требования, капсула задачи, реализация, архитектура, тестирование, прослеживаемость, выпуск
Усиленный	Безопасность, данные, публичные интерфейсы, миграции, критичные сервисы	Полный набор ролей, моделирование угроз, репетиция выпуска, откат, окно наблюдения и разбор возможного инцидента

Это соответствует принципу AWS: маршрут и глубина работы должны зависеть от намерения, масштаба и риска.

6. Неопределённость остаётся видимой

Если данных нет, источник устарел или проверка охватывает только часть требования, система показывает это явно. Статусы «частично», «не подтверждено» и «заблокировано» полезнее зелёного индикатора, который скрывает пробел.

Так команда видит разницу между отсутствием проблемы и отсутствием сведений о проблеме.

7. Выпуск запускает проверку продуктового результата

После развёртывания CodeGraph связывает изменение с наблюдаемостью, инцидентами, DORA-метриками, стоимостью и продуктовым результатом. Эти данные становятся входом для следующего решения: расширить изменение, скорректировать его, откатить или остановить развитие гипотезы.

2. Основные объекты плейбука

Продуктовая цель

Кратко описывает проблему, пользователя, ожидаемое изменение и показатель результата. Цель остаётся стабильной точкой, относительно которой оцениваются требования и фактический эффект.

Требования в OpenSpec

OpenSpec хранит версионируемое описание того, как должна вести себя система. В нём фиксируются:

- границы изменения;
- функциональные и нефункциональные требования;
- ограничения;
- критерии приёмки;
- связи с продуктовой историей и задачами.

Каждый критерий получает идентификатор. На него ссылаются тесты, проверки и решения.

Капсула задачи

Капсула превращает часть требований в управляемую работу. Она отвечает на семь вопросов:

1. Какова цель этой задачи?
2. Какой контекст нужен исполнителю?
3. Что входит и что не входит в работу?

4. Когда задача считается выполненной?
5. Кто отвечает за результат?
6. Какие проверки обязательны?
7. С какими требованиями и материалами связана задача?

Капсула удерживает действие агента внутри согласованной цели.

Передача результата

При переходе между ролями передаётся содержательный пакет:

- выполненная работа;
- ссылки на исходные материалы;
- результаты проверок;
- известные ограничения;
- открытые вопросы;
- следующая ответственная роль.

Получатель проверяет пакет в своей области и формирует собственное заключение.

Пакет подтверждений

Пакет подтверждений связывает требование с конкретной версией системы. Он содержит:

- идентификатор требования и критерия;
- ревизию кода и конфигурации;
- способ проверки;
- результат;
- время;
- ответственную роль;
- оставшийся риск;
- решение человека.

Кандидат на выпуск

Кандидат — это точная комбинация кода, сборки, конфигурации и целевой среды. Изменение любого из этих элементов создаёт новый кандидат и обновляет связанные проверки.

3. Сквозной цикл CodeGraph

Этап 1. Сформулировать продуктовый результат

Что происходит

Менеджер продукта собирает исходные сигналы: обращения пользователей, данные, стратегические цели, дефекты и эксплуатационные события. Он формулирует наблюдаемый результат и задаёт границы задачи.

Системный аналитик переводит замысел в требования и критерии приёмки. Требования не дублируют друг друга и вместе покрывают заявленную область.

Что фиксирует CodeGraph

- продуктовую цель;
- пользователя или оператора;
- ожидаемый результат;
- основную метрику;
- ограничивающие показатели;
- требования и критерии приёмки;
- владельцев решений.

Условие перехода

Команда понимает, что должно измениться и как это будет проверено.

Основные роли

Кира отвечает за продуктовую постановку. **Сергей** отвечает за требования, ограничения и критерии приёмки.

Этап 2. Создать капсулу задачи и выбрать маршрут

Что происходит

Руководитель поставки выделяет исполнимый срез требований. Он определяет зависимости, сроки, участников, обязательные проверки и последовательность передачи результата.

Затем выбирается маршрут: быстрый, основной или усиленный. На выбор влияют:

- критичность сервиса;
- работа с данными и правами;
- изменение публичных интерфейсов;
- обратимость;
- объём затронутого кода;
- качество тестов;
- сложность выпуска.

Что фиксирует CodeGraph

- капсулу задачи;
- связь с требованиями;
- область разрешённых изменений;
- назначенные роли;
- обязательные проверки;
- предварительную оценку затрат;
- условия остановки и эскалации.

Готовые сценарии

Маршрут собирается из ограниченных сценариев CodeGraph: знакомство с кодовой базой, разработка функции, рефакторинг, архитектурная проверка, аудит безопасности, поиск пробелов в тестах, анализ зависимостей и расследование инцидента. У каждого сценария есть входные данные, разрешённые инструменты, формат результата и обязательные проверки. Полный перечень приведён в каталоге сценариев.

Условие перехода

У исполнителя есть понятная задача, доступ к нужному контексту и разрешение на выбранные действия.

Основные роли

Анна отвечает за план, зависимости и маршрут. **Марина** отвечает за настройки цифровых ролей и их полномочия. **Ева** формирует предварительную оценку затрат.

Этап 3. Спроектировать проверку до реализации

Что происходит

До изменения кода команда определяет, какими признаками будет подтверждён результат.

Тестировщик готовит позитивные, негативные, граничные и восстановительные сценарии. Специалист по безопасности описывает проверки прав, данных, секретов и внешних границ. Архитектор оценивает новые интерфейсы и область возможного влияния. Ответственный за прослеживаемость связывает каждый критерий с ожидаемым подтверждением.

Что фиксирует CodeGraph

- тестовый план;
- требования безопасности;
- архитектурные ограничения;
- карту «критерий → проверка → ожидаемый результат»;
- список обязательных заключений;
- план отката для рискованных изменений.

Условие перехода

Для каждого важного критерия известен способ проверки.

Основные роли

Лена отвечает за сценарии приёмки. **Вера** отвечает за безопасность приложений. **Рита** отвечает за архитектурную проверку. **Иван** отвечает за связи между требованиями, тестами и результатами.

Этап 4. Выполнить минимальное изменение

Что происходит

Разработчик или ИИ-агент сначала воспроизводит отсутствующее или ошибочное поведение тестом. Затем вносит минимальное изменение, которое приводит тест к успеху, и запускает затронутую регрессию.

Такой порядок удерживает работу внутри капсулы задачи и даёт проверяемую причину изменения.

Что фиксирует CodeGraph

- базовую и итоговую ревизии;
- изменённые файлы и символы;
- выполненные команды;
- результаты тестов;
- неподтверждённые предположения;
- сведения об исполнителе и применённом сценарии.

Условие перехода

Изменение решает заявленный технический срез, а локальные проверки относятся к той же ревизии.

Основная роль

Дмитрий отвечает за реализацию и связанные тесты.

Этап 5. Провести независимые проверки

Что происходит

После реализации включаются отдельные контуры.

Архитектура

Рита проверяет зависимости, общие компоненты, новые интерфейсы и область влияния.

Безопасность

Вера проверяет права доступа, секреты, потоки данных, защиту от утечек и открытые риски.

Тестирование

Лена сопоставляет поведение системы с критериями приёмки и выполняет нужные профили тестов.

Документация

Борис обновляет пользовательские, технические и эксплуатационные материалы.

Прослеживаемость

Иван проверяет цепочку:

цель → требование → задача → изменение → тест → подтверждение → выпуск

Эксплуатация

Глеб оценивает готовность среды, наблюдаемость, развёртывание и откат.

Каждая роль формирует отдельное заключение. Сводка сохраняет исходные результаты и показывает общий статус.

Условие перехода

Все обязательные роли дали заключения, а открытые риски получили владельца и решение.

Этап 6. Подготовить и выполнить выпуск

Что происходит

Команда собирает точного кандидата на выпуск и повторяет обязательные проверки относительно его кода, сборки и конфигурации.

Перед решением о выпуске фиксируются:

- ревизия кода;
- версия и контрольная сумма сборки;
- конфигурация;
- список изменений;
- миграции;
- целевая среда;
- окно выпуска;
- ответственный за откат;
- проверки после развёртывания.

CodeGraph формирует матрицу готовности по продукту, реализации, архитектуре, безопасности, тестированию, документации, прослеживаемости, эксплуатации и затратам.

Техническая проверка выпуска отвечает на вопрос: **готов ли этот кандидат к развёртыванию по выбранному профилю?**

Итоговое закрытие отвечает на более широкий вопрос: **закрыты ли все продуктовые критерии, обязательные заключения, затраты и решения по рискам?**

После развёртывания

Глеб проверяет:

- работоспособность сервиса;
- миграции;
- критические пользовательские сценарии;
- телеметрию;
- пороги отката;
- состояние системы в согласованном окне наблюдения.

Условие перехода

Кандидат выпущен, наблюдается, а ответственные приняли все обязательные решения.

Этап 7. Вернуть результат в продуктовый цикл

Что происходит

После выпуска команда сопоставляет исходную гипотезу с фактом.

Она рассматривает четыре группы данных:

1. **Продукт:** изменилось ли поведение пользователя или бизнеса.
2. **Поставка:** как изменились скорость и устойчивость разработки.
3. **Качество:** сколько дефектов, рисков и возвратов на доработку возникло.
4. **Экономика:** сколько стоил принятый результат, включая работу людей и ИИ.

Результатом становится одно из четырёх решений:

- расширить изменение;
- скорректировать продукт или процесс;
- ограничить область применения;
- прекратить развитие гипотезы.

Инциденты

Если после выпуска возник инцидент, CodeGraph связывает его с сервисом, средой, выпуском и ревизией. Работа проходит по короткому циклу:

обнаружить
→ оценить влияние
→ стабилизировать
→ восстановить
→ установить причину
→ провести исправление через обычный процесс
→ проверить эффективность меры

Корректирующие меры получают владельца, срок, связанное требование и дату проверки эффективности.

Условие завершения цикла

Продуктовый владелец принял решение на основе результата, качества, стоимости и эксплуатационных данных.

4. Цифровая команда CodeGraph

Тринадцать ролей покрывают полный цикл. Для небольшого пилота часть ролей может выполнять один человек, но области ответственности остаются раздельными.

Контур	Цифровая роль	Результат
Продукт	Кира, менеджер продукта	Цель, пользовательский результат и границы
Требования	Сергей, системный аналитик	Требования и критерии приёмки
Поставка	Анна, руководитель поставки	Капсула задачи, план, зависимости и общий статус
Реализация	Дмитрий, разработчик	Изменение кода и тесты
Архитектура	Рита, архитектурная проверка	Область влияния, нарушения и архитектурное заключение
Безопасность	Вера, безопасность приложений	Угрозы, проверки и открытые риски
Качество	Лена, тестирование	Результаты приёмочных и регрессионных тестов
Документация	Борис, документация и знания	Обновлённые материалы и инструкции
Прослеживаемость	Иван, связь требований с реализацией	Найденные и отсутствующие связи
Поддержка	Олег, поддержка	Проверенные инструкции диагностики и восстановления
Управление ИИ	Марина, управление цифровыми ролями	Версии ролей, инструкции, полномочия и контроль изменений
Эксплуатация	Глеб, эксплуатация и выпуск	Готовность среды, развёртывание, наблюдение и откат
Экономика	Ева, учёт затрат	Время, токены, стоимость и распределение затрат

Главное правило этой модели: итоговый статус складывается из независимых результатов всех обязательных ролей.

5. Как CodeGraph понимает изменение кода

Граф свойств кода

Граф свойств кода, или CPG, соединяет структуру программы в одну модель:

- файлы;
- функции и классы;
- вызовы;
- зависимости;
- точки входа;
- потоки данных;
- связанные тесты.

Когда команда меняет функцию, CodeGraph может показать:

- кто её вызывает;
- какие сервисы и интерфейсы зависят от результата;
- через какие компоненты проходят данные;
- какие тесты относятся к затронутой области;
- где появились новые риски;
- какие связи не удалось определить автоматически.

Обычный поиск отвечает на вопрос «где встречается это имя». Граф помогает ответить на вопрос «что может измениться вместе с этим кодом».

Пример

Требование: роль администратора должна проверяться до изменения лимита.

CodeGraph связывает:

```
требование SEC-ADM-014
→ изменение маршрутизатора и сервиса политик
→ функции проверки роли
→ тесты административных сценариев
→ библиотеку аудита
→ открытый риск отсутствия владельца библиотеки
→ решение архитектора
```

Так продуктовый критерий получает инженерное подтверждение, а найденный риск — конкретного владельца.

Состояния требования

Состояние	Что оно означает
Реализовано	Есть связанное изменение, обязательные проверки пройдены
Частично	Подтверждена только часть требования
Не подтверждено	Не хватает связи с кодом, тестом или свежим результатом
Заблокировано	Открыт критичный риск или обязательная проверка не пройдена

6. Три уровня контроля

1. Полномочия

Полномочия определяют, какие действия доступны роли:

```
читать
→ готовить предложение
→ менять код в ветке
→ создавать запрос на слияние
→ объединять изменение после проверок
→ развёртывать в тестовой среде
→ выполнять ограниченные операции в промышленной среде
```

Уровень зависит от риска, обратимости, тестового покрытия, типа данных и критичности системы.

2. Подтверждения

Для каждого результата сохраняются:

- источник;
- версия;

- время;
- способ проверки;
- заключение роли;
- ограничения;
- решение человека.

Удобно различать четыре уровня зрелости результата:

Уровень	Пример	Назначение
Рабочий сигнал	журнал, локальный запуск, черновик агента	помогает в работе
Воспроизводимый результат	тест, отчёт, контрольная сумма, снимок состояния	подтверждает отдельный критерий
Независимое заключение	архитектура, безопасность, тестирование, эксплуатация	закрывает область ответственности
Принятое состояние	итог по всем обязательным критериям и решениям	разрешает переход процесса

3. Решение

Человек принимает решения, которые меняют:

- продуктовую цель;
- границы требования;
- архитектурный риск;
- исключение из политики;
- выпуск;
- закрытие задачи;
- бюджет и масштабирование.

ИИ готовит материал, выявляет пробелы и выполняет повторяемые действия. Решение сохраняется вместе с областью действия и сроком.

7. Безопасная разработка и соответствие требованиям

CodeGraph связывает требования безопасности с кодом, проверками, рисками и решениями. Для каждой применимой области создаётся простая цепочка:

```
обязательство
→ контроль
→ проверяемый материал
→ ответственное лицо
→ версия
→ результат проверки
```

Платформа ведёт карту 25 процессных областей ГОСТ Р 56939-2024: от планирования и моделирования угроз до безопасной сборки, управления зависимостями, выпуска, реагирования на уязвимости и вывода системы из эксплуатации.

Вместо большой таблицы документов команда формирует матрицу готовности:

Поле	Содержание
Требование	Какое обязательство выполняется
Контроль	Какой организационный или технический механизм действует

Поле	Содержание
Материал	Что можно проверить повторно
Ответственный	Кто принимает результат или риск
Версия	К какой версии системы относится проверка
Состояние	Выполнено, действует компенсирующая мера или остаётся пробел
Проверка	Когда и каким способом подтверждён результат

Для внешней оценки CodeGraph экспортирует связанный пакет: состав кандидата, контрольные суммы, матрицу требований и контролей, журнал действий и результаты проверок.

Подробности собраны в материалах CodeGraph по безопасности, доказательствам выполнения требований и проверке готовности к выпуску.

8. Что измерять

Продуктовый результат

Главный вопрос: достигло ли изменение цели, ради которой его начали.

Примеры:

- доля успешно завершённых пользовательских сценариев;
- время выполнения операции;
- конверсия;
- число обращений в поддержку;
- стоимость бизнес-процесса;
- доля ошибок пользователя.

Поток поставки

CodeGraph связывает события из репозиториев, выпусков, развёртываний и инцидентов. В текущей модели используются четыре показателя DORA:

- частота развёртываний;
- время прохождения изменения до промышленной среды;
- доля выпусков с отказом;
- время восстановления.

Метрики DORA показывают скорость и устойчивость поставки. Их нужно рассматривать вместе с продуктовым результатом и качеством. Официальное описание показателей доступно в руководстве DORA.

Качество

- дефекты после выпуска;
- критичность найденных проблем;
- возвраты на доработку;
- время проверки;
- архитектурные нарушения;
- непокрытые тестами участки;

- доля частично подтверждённых требований.

Экономика

CodeGraph связывает затраты с задачей, ролью и принятым результатом.

Базовые показатели:

```
Стоимость принятого результата =
общая стоимость работы / число принятых результатов

Стоимость переделки =
затраты ИИ и людей после отклонения первого варианта

Доля нераспределённых затрат =
затраты без точной связи с задачей / все затраты

Экономический эффект =
подтверждённая выгода − стоимость поставки − стоимость инцидентов
```

Расход токенов сам по себе не показывает ценность. Он становится полезным, когда связан с принятым результатом, ручным временем, качеством и сроком.

9. Пилот: подготовка и шесть рабочих недель

Плейбуки AWS, Сбера и Т-Банка рекомендуют начинать с ограниченного сценария и расширять охват после измерения результата.

Подготовительная неделя. Границы и исходные показатели

Выберите один сервис, одну команду и один повторяемый тип изменения. Зафиксируйте:

- время прохождения задачи;
- ручные часы;
- время проверки;
- дефекты;
- возвраты на доработку;
- показатели DORA;
- стоимость;
- качество источников данных.

Неделя 1. Продукт и требования

Проведите одну реальную задачу от продуктовой цели до OpenSpec. Сформулируйте измеримый результат и проверяемые критерии приёмки.

Неделя 2. Капсула и реализация

Создайте капсулу задачи, назначьте маршрут и роли. Реализуйте минимальное изменение через тест, который сначала воспроизводит отсутствующее поведение.

Неделя 3. Независимые проверки

Подключите архитектуру, безопасность, тестирование, документацию и прослеживаемость. Сопоставьте результаты с точной ревизией.

Неделя 4. Выпуск и наблюдение

Соберите кандидата, проверьте развёртывание и откат, проведите выпуск и наблюдайте систему в согласованном окне.

Неделя 5. Репетиция инцидента

Проведите учебный сценарий: обнаружение, стабилизация, восстановление, связь с выпуском, разбор причин и проверка корректирующей меры.

Неделя 6. Экономика и решение

Сравните пилот с исходными показателями. Отдельно оцените:

- продуктовый эффект;
- скорость поставки;
- качество;
- нагрузку на людей;
- стоимость;
- качество собранных данных.

Результат пилота — решение расширить сценарий, изменить процесс, сократить область или остановить внедрение.

10. Как CodeGraph соотносится с другими плейбуками

Плейбук	Главная идея	Как она отражена в CodeGraph
Anthropic: плейбук разработки с ИИ-агентами	Каждая стадия заканчивается машиночитаемым артефактом, а эксплуатация возвращает данные в новый цикл	OpenSpec, капсула задачи, пакет подтверждений, кандидат на выпуск и связь с инцидентами
AWS: жизненный цикл разработки с ИИ	ИИ планирует, уточняет и исполняет; люди принимают критические решения	Цифровые роли выполняют ограниченную работу, а переходы и риски утверждают ответственные сотрудники
AWS: адаптивные маршруты разработки	Состав и глубина стадий меняются под задачу	Быстрый, основной и усиленный маршруты; набор ролей и проверок зависит от риска
Сбер: AI-Disrupt PDLC	Контур намерения отделён от исполнения, а проверка и управление проходят через весь цикл	Продуктовая цель и требования отделены от реализации; полномочия и подтверждения действуют на каждом этапе
Т-Банк: AI4SDLC	Общая платформа, портфель сценариев, пилоты и командные метрики	Каталог ограниченных сценариев, цифровые роли, пилот на одном сервисе и измерение потока поставки
CodeGraph	Связать продуктовый замысел с фактической структурой кода и решением о выпуске	Граф свойств кода, сквозная прослеживаемость, независимые заключения и единый пакет готовности

CodeGraph делает видимыми связи, которые команда обычно восстанавливает вручную перед проверкой, выпуском или аудитом.

11. Минимальные шаблоны

Капсула задачи

Цель:
Контекст:
Что входит в работу:
Что не входит:
Готово, когда:
Ответственный:
Обязательные проверки:
Допустимые исключения:
Связанные требования и критерии:
Связанные задачи и материалы:

Карточка критерия приёмки

Идентификатор:
Проверяемое утверждение:
Связанные требования:
Ревизия:
Способ проверки:
Результат:
Заключение ответственной роли:
Решение человека:
Оставшийся риск:

Решение о выпуске

Кандидат:
Ревизия кода:
Версия и контрольная сумма сборки:
Конфигурация:
Среда и окно выпуска:
Входящие изменения:
Пройденные проверки:
Принятые риски и владельцы:
Условие и способ отката:
Проверки после выпуска:
Решение:
Ответственный и время:

Мера после инцидента

Изменение:
Связь с причиной:
Ответственный:
Срок:
Связанное требование и тест:
Целевой выпуск:
Дата проверки эффективности:
Измеримый показатель:
Состояние:

12. Проверка готовности процесса

- ☐ Продуктовый результат измерим.
- ☐ Требования и критерии приёмки имеют версии и владельцев.
- ☐ Для работы создана капсула задачи.
- ☐ Маршрут соответствует риску изменения.
- ☐ У каждого важного критерия есть способ проверки.
- ☐ Реализация связана с точной ревизией и тестами.

- Архитектура, безопасность и тестирование дали независимые заключения.
- Документация и прослеживаемость обновлены.
- Кандидат на выпуск имеет точную идентичность.
- Развёртывание, наблюдение и откат проверены.
- Открытые риски имеют владельцев и решения.
- Затраты связаны с задачами и принятыми результатами.
- После выпуска собран продуктовый и эксплуатационный результат.
- Итоговый статус принят назначенным сотрудником.

13. Что организация настраивает под себя

Плейбук задаёт общую модель. Конкретная компания определяет восемь правил.

1. Основной источник требований

Нужно выбрать, где хранится исходная версия: в OpenSpec, трекере, системе управления требованиями или другом корпоративном инструменте. Остальные системы содержат ссылку или рабочую копию.

2. Классы изменений и маршруты

Компания определяет, какие изменения проходят быстрый, основной или усиленный маршрут и какие признаки повышают уровень контроля.

3. Уровни самостоятельности цифровых ролей

Для каждой роли задаются доступные операции: чтение, предложение, изменение в ветке, создание запроса на слияние, объединение, развёртывание и действия в промышленной среде.

4. Обязательные человеческие решения

Фиксируются сотрудники и заместители, которые утверждают требования, архитектурный риск, исключения по безопасности, выпуск и бюджет.

5. Набор проверок

Для каждого типа изменения определяется минимальный набор тестов, архитектурных правил, требований безопасности, документации и эксплуатационных проверок.

6. Актуальность данных

Источники получают владельца и допустимый срок обновления. Устаревшие сведения отображаются отдельно от свежих результатов.

7. Контракты интеграций

Для Git, трекера, CI/CD, развёртывания, наблюдаемости и учёта затрат задаются идентификаторы связи, правила повторной обработки и порядок восстановления после рассинхронизации.

8. Продуктовая обратная связь

Компания определяет, какая метрика подтверждает результат, когда она измеряется и кто принимает следующее продуктовое решение.

14. Основные материалы

CodeGraph

- Архитектура CodeGraph: от задачи до выпуска
- Граф свойств кода
- Каталог сценариев
- Документация CodeGraph
- Цифровая команда: роли и полномочия
- Доказательства выполнения требований
- Проверка требований безопасности
- Проверка готовности к выпуску

Другие плейбуки и методики

- Anthropic: плейбук разработки с ИИ-агентами
- AWS: жизненный цикл разработки с ИИ
- AWS: адаптивные маршруты разработки
- Сбер: AI-Disrupt PDLC
- Т-Банк: AI4SDLC
- Т-Банк: обзор практик ИИ в программной инженерии
- DORA: показатели эффективности поставки ПО

Итог

Плейбук CodeGraph строится вокруг одного объекта: принятого изменения, связь которого можно проследить от продуктовой цели до поведения системы после выпуска.

ИИ-агенты ускоряют подготовку требований, анализ, реализацию и проверку. CodeGraph соединяет их работу с ролями, полномочиями, графом кода, тестами, рисками и решениями. В результате команда управляет скоростью, качеством, стоимостью и предсказуемостью поставки.